UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/840,727	04/23/2001	Alando M. Ballantyne	014208.1360	4537

35005 7590 03/30/2004

BAKER BOTTS L.L.P.
2001 ROSS AVENUE, 6TH FLOOR
DALLAS, TX 75201

EXAMINER	
TANG, KUO LIANG J	

ART UNIT	PAPER NUMBER
2122	5

DATE MAILED: 03/30/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/840,727	Applicant(s) BALLANTYNE ET AL.	
	Examiner Kuo-Liang J Tang	Art Unit 2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE ____ MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *06 January 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-23 and 25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-23 and 25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the application filed on 01/06/2004.

Claims 1-23 and 25 are pending and the effective date is 03/09/2000.

Claims 7-9, 23 and 25 remain rejected under 35 U.S.C. 102(e). (Note that claims 23 and 25 are amended to correct minor informalities, and the amendment does not change the scope of these claims. Therefore, the same rejections set forth in the Office action mailed on 10/16/2003 also applies hereto with minor augmentation in response to the claims now amended.)

Claims 1-6 and 10-22 remain rejected under 35 U.S.C. 103(a). (Note that claims 3 and 16 are amended to correct minor informalities, and the amendment does not change the scope of these claims. Therefore, the same rejections set forth in the Office action mailed on 10/16/2003 also applies hereto with minor augmentation in response to the claims now amended.)

Claim Rejections - 35 USC § 102

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

2. Claims 7-9 are rejected under 35 U.S.C. 102(e) as being anticipated by Lection et al. (US Patent No. 6,418,446) hereafter Lection.

As Per Claim 7, Lection disclosed:

A method for outputting data from an application running on a computer system, the data output as Extensible Markup Language, the method comprising:

-establishing a relationship of the output data and one ... Extensible Markup Language Document Object Model contexts; (see Column 3, Lines 8-10, "It is another object of the present invention to provide this technique using a DOM tree created from an XML syntax representation of the source data,"). DOM tree is the relationship of the output data. The relationship must have been established in the DOM tree, otherwise the DOM tree will be inoperable (E.g. see FIG. 3D and associated text.).

-building a Document Object Model instance with the one... contexts; (see Column 3, Lines 10-12, "creating an output DOM tree in which the destination data gathered from the source is reformatted and stored.") and (see Column 3, Lines 34-39, "This technique comprises: providing an input data source comprising one or more records, wherein each of the records has this dynamically variable record format, and wherein the dynamically variable record format of each record comprises a plurality of dynamically variable fields;") and (see Column 3, Lines 52-56, "Optionally, the first DOM tree may be created by parsing an Extended Markup Language (XML) representation of the selected record and the second DOM tree may be created by parsing an XML representation of the gather verb specification.") (Emphasis added) Examiner interprets the representation of the selected record and/or representation of the gather verb specification is context; and

-outputting the data from the Document Object Model instance as Extensible Markup Language. (see Column 3, Lines 48-52, "The selected record may be formatted as a first

Document Object Model (DOM) tree, the gather verb specification may be formatted as a second DOM tree, and the output data destination may be formatted as a third DOM tree.").

As Per Claim 8, the rejection of claims 7 is incorporated and further Lection disclosed:

-activating plural contexts simultaneously to buffer data for output as a complete Document Object Model instance. (see Column 3, Lines 34-39, "This technique comprises: providing an input data source comprising one or more records, wherein each of the records has this dynamically variable record format, and wherein the dynamically variable record format of each record comprises a plurality of dynamically variable fields;") and (see Column 3, Lines 52-56, "Optionally, the first DOM tree may be created by parsing an Extended Markup Language (XML) representation of the selected record and the second DOM tree may be created by parsing an XML representation of the gather verb specification.")

As Per Claim 9, the rejection of claims 8 is incorporated and further Lection disclosed:

-creating a node for an output data; and ensuring the correct cardinality of the created node. (see Column 3, Lines 48-52, "The selected record may be formatted as a first Document Object Model (DOM) tree, the gather verb specification may be formatted as a second DOM tree, and the output data destination may be formatted as a third DOM tree.").

3. Claims 23-25 are rejected under 35 U.S.C. 102(e) as being anticipated by Stefaniak (US Patent No. 6,550,054).

As Per Claim 23, Stefaniak discloses a method for modeling a legacy computer system comprising:

"identifying incidents of applications of the legacy computer system that output data;" (E.g. see Column 1, Lines 41-47).

"associating the incidents with an Extensible Markup Language schema;" (E.g. see Column 2, Lines 48-51 and FIG.9 and associated text). and

"defining a control flow graph of the output incidents;" (E.g. see FIG. 3 and associated text; Column 2, Lines 29-30) and

"creating a specification to modify the legacy computer system applications to provide output from a Document Object Model instance as Extensible Markup Language." (E.g. see Column 2, Lines 39-45.").

"automatically modifying the legacy computer system applications in accordance with the specification." (E.g. see Column 1, Lines 58-67, which states "... The method includes the steps of transforming a terminal-based screen application into an application specification; ..." (emphasis added) and see FIG. 4, project 46, create or edit file and associated text). Examiner disagrees with applicant's assertion that "there is no modification of the terminal-based application in Stefaniak". Stefaniak discloses a system that describes legacy application screens in term of a terminal application specification and converts the specification into a UML model (E.g. see Abstract and col. 1:57-67). Stefaniak also discloses using the transform navigator 19, which produces application and screen specification. (E.g. see col. 5:41-45). As showing in FIG. 4, one of the transform processes is the project 46, which is "create/edit file reference model for project" that produce the final result display 48 for an end user 49 to view. (Note that edit is

known as "to alter, adapt"; modify is known as "to make change"; convert is known as "to change from from one form or function to another"; transform is known as "to change in composition or structure", see "Microsoft Computer Dictionary", fifth edition, pages 367, 748, 253 and 1253 respectively).

As Per Claim 25, Stefaniak disclosed:A system for modeling an output application of a legacy computer system comprising:

"a modeling engine interfaced with the legacy computer system, the modeling engine operable to analyze an application loaded on the legacy computer system to identify incidents within the application that output data from the legacy computer system;" (E.g. see Column 1, Lines 41-47).

"a control flow graph of the output operations within the applications, the control flow graph having plural nodes, each node associated with an output incident;" (E.g. see FIG. 3 and associated text; Column 2, Lines 29-30; Column 5, Lines 39-57). and

"a graphical user interface in communication with the modeling engine, the graphical user interface operable to display the control flow graph and the incidents; wherein the graphical user interface maps the incidents of the applications with the control flow graph and an Extensible Markup Language schema." (E.g. see Column 1, Lines 58-67). and

"a code generation engine in communication with the graphical user interface and the legacy computer system, the code generation engine operable to modify legacy computer system application code to directly output data from a Document Object Model as Extensible Markup Language." The code generation engine is inherent because in order for an end user to view the

display, a code generation engine must be there otherwise it will not work (E.g. see FIG.4 and associated text).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-2, 4-6, 20-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stefaniak (US Patent No. 6,550,054) in view of van Elkeren et al. (US Patent No. 6,618,852) hereafter van Elkeren.

As Per Claim 1, Stefaniak disclosed: A method for reporting data from a legacy computer system using Extensible Markup Language, the method comprising:

-generating a model of the legacy computer system; (see Column 1, Lines 58-67, "These and other objects, which will become apparent as the invention is described in detail below, are provided in a computer-implemented method that automatically converts text-based screen applications of a legacy computer system into a graphical-based representation thereof. The method includes the steps of transforming a terminal-based screen application into an application specification; converting the application specification into a modeling language-based

representation; and, displaying the modeling language-based representation with a graphical user interface.").

-mapping the model of the legacy computer system to an Extensible Markup Language schema; (see Column 1, Line67 to column 2, lines 1-4 , "This method also includes the capability of generating document type definitions of the modeling language-based representation, which enables transmission of the representation among modeling tools."). "document type definitions" is XML schema. and

-automatically modifying one or more applications of the legacy computer system, (see Column 5, Lines 39-43, "Referring now to FIG. 3, a diagram that shows an end-to-end process flow from a legacy program to an XML/UML file is shown. The process flow begins with any terminal based legacy application 30, which produces application terminal screens 31") the modified application operable to output data written from the legacy computer system in Extensible Markup Language. (see Column 5, Lines 43-57, "The terminal screens are discovered using the transform navigator 19, which produces application and screen specifications 32. The application and screen specifications 32 are then applied to the file warehouse 21, which produces the project file reference model 27. The model 27 is applied to the terminal-to-XML 20, which produces a UML model 34 in a MOF compliant repository. The UML model is then applied to the UML model to XMI/UML DTD generator 22, which produces XMI/UML DTD streams 35. The streams 35 may be used for several purposes, including transmitting legacy screen based application specifications over a network to modeling tools. From the modeling tools the application specifications could be viewed in an object oriented way compliant with the UML standard.").

Stefaniak didn't explicitly disclose using a Document Object Model. However, van Elkeren taught automatically modifying one or more applications of the legacy computer system, the modified application operable to output data written using a Document Object Model from the legacy computer system in Extensible Markup Language. (see Column 12, Lines 5-10, "Preferably, a standard API, the XML DOM (Document Object Model) is used to access and manipulate XML data. The DOM provides a standard set of properties, methods, and events for program developers to use. This set of standards, the object model, provides a simple means of reading and writing data to and from an XML tree structure."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use DOM, as suggested by van Elkeren, to apply to output XML data. The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide a simple means of reading and writing data to and from an XML tree structure.

As Per Claim 2, the rejection of claim 1 is incorporated and further Stefaniak disclosed:

-providing the legacy computer system with a writer engine, the writer engine having the Extensible Markup Language Schema loaded as a data file; (see Column 1, Line67 to column 2, lines 1-4 , "This method also includes the capability of generating document type definitions of the modeling language-based representation, which enables transmission of the representation among modeling tools."). "document type definitions" is XML schema. and

Stefaniak didn't explicitly disclose populating a Document Object Model. However, van Elkeren taught calling the writer engine with the modified applications, the writer engine populating the Document Object Model according to the Extensible Markup Language schema

by building a Document Object Model instance with one or more contexts. (see Column 12, Lines 5-10, "Preferably, a standard API, the XML DOM (Document Object Model) is used to access and manipulate XML data. The DOM provides a standard set of properties, methods, and events for program developers to use. This set of standards, the object model, provides a simple means of reading and writing data to and from an XML tree structure."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to populate DOM, as suggested by van Elkeren, to apply to output XML data. The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide a simple means of reading and writing data to and from an XML tree structure.

As Per Claim 4, Stefaniak disclosed: A system for reporting data from a legacy computer system in an Extensible Markup Language format, the system comprising:

-a modeling engine in communication with the legacy computer system, the modeling engine operable to generate a model of reported data written by an application residing on the legacy computer system; (see Column 1, Lines 58-67, "These and other objects, which will become apparent as the invention is described in detail below, are provided in a computer-implemented method that automatically converts text-based screen applications of a legacy computer system into a graphical-based representation thereof. The method includes the steps of transforming a terminal-based screen application into an application specification; converting the application specification into a modeling language-based representation; and, displaying the modeling language-based representation with a graphical user interface.").

-a mapping engine in communication with the modeling engine, the mapping engine operable to generate a modification specification by mapping the model to an Extensible Markup Language schema; (see Column 5, Lines 43-57, "The terminal screens are discovered using the transform navigator 19, which produces application and screen specifications 32. The application and screen specifications 32 are then applied to the file warehouse 21, which produces the project file reference model 27. The model 27 is applied to the terminal-to-XML 20, which produces a UML model 34 in a MOF compliant repository. The UML model is then applied to the UML model to XMI/UML DTD generator 22, which produces XMI/UML DTD streams 35. The streams 35 may be used for several purposes, including transmitting legacy screen based application specifications over a network to modeling tools. From the modeling tools the application specifications could be viewed in an object oriented way compliant with the UML standard."). and

-a code generation engine in communication with the mapping engine and the legacy computer system, the code generation engine operable to modify legacy computer system application code to directly output data from Extensible Markup Language. (see Column 5, Lines 43-57, "The terminal screens are discovered using the transform navigator 19, which produces application and screen specifications 32. The application and screen specifications 32 are then applied to the file warehouse 21, which produces the project file reference model 27. The model 27 is applied to the terminal-to-XML 20, which produces a UML model 34 in a MOF compliant repository. The UML model is then applied to the UML model to XMI/UML DTD generator 22, which produces XMI/UML DTD streams 35. The streams 35 may be used for several purposes, including transmitting legacy screen based application specifications over a network to modeling

tools. From the modeling tools the application specifications could be viewed in an object oriented way compliant with the UML standard.").

Stefaniak didn't explicitly disclose using a Document Object Model. However, van Elkeren taught a code generation engine in communication with the mapping engine and the legacy computer system, the code generation engine operable to modify legacy computer system application code to directly output data from a Document Object Model as Extensible Markup Language. (see Column 12, Lines 5-10, "Preferably, a standard API, the XML DOM (Document Object Model) is used to access and manipulate XML data. The DOM provides a standard set of properties, methods, and events for program developers to use. This set of standards, the object model, provides a simple means of reading and writing data to and from an XML tree structure."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use DOM, as suggested by van Elkeren, to apply to output XML data. The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide a simple means of reading and writing data to and from an XML tree structure.

As Per Claim 5, the rejection of claim 4 is incorporated and further Stefaniak disclosed:

-a context table associated with the legacy computer system, the context table providing the Extensible Markup Language schema to the legacy computer system; and

-a writer engine loaded on the legacy computer system and having the Extensible Markup Language schema stored as a data file, the writer engine communicating with the modified

legacy computer system applications to buffer data in plural contexts for output as Extensible Markup Language. (see Column 6, Lines 41-58, "Referring now to FIG. 5C, at the connector C, a UML attribute and a UML datatype tag are assigned for each field name associated with the screen (block 65). Next, all the operations of the screen are initialized and transmitted as the operations of the UML class created for that screen (block 66). This is followed by making an inquiry whether or not there are any more screens in the UML package (block 67). If the answer to this enquiry is yes, then a return is make back to block 62 (FIG. 5B) as denoted by a connector E. If on the other hand, the answer to the above inquiry is no then a further inquiry is made to determine whether or not there are any more UML packages (block 68). If the answer to this inquiry is yes, then a return is made back to the block 61 (FIG. 5B) as denoted by a connector D. If on the other hand, the answer to the above inquiry is no, then a step of saving the tempfile as tempfile.txt is executed (block 69). After this, the process ends (bubble 70).").

Stefaniak didn't explicitly disclose using a Document Object Model. However, van Elkeren taught a writer engine loaded on the legacy computer system and having the Extensible Markup Language schema stored as a data file, the writer engine communicating with the modified legacy computer system applications to buffer data in plural contexts within a Document Object Model for output as Extensible Markup Language. (see Column 12, Lines 5-10, "Preferably, a standard API, the XML DOM (Document Object Model) is used to access and manipulate XML data. The DOM provides a standard set of properties, methods, and events for program developers to use. This set of standards, the object model, provides a simple means of reading and writing data to and from an XML tree structure."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use DOM, as

suggested by van Elkeren, to apply to output XML data. The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide a simple means of reading and writing data to and from an XML tree structure.

As Per Claim 6, the rejection of claim 5 is incorporated and further Stefaniak disclosed:

- the writer engine is coded in the computer language of the legacy computer system. (see Column 6, Lines 41-58, "Referring now to FIG. 5C, at the connector C, a UML attribute and a UML datatype tag are assigned for each field name associated with the screen (block 65). Next, all the operations of the screen are initialized and transmitted as the operations of the UML class created for that screen (block 66). This is followed by making an inquiry whether or not there are any more screens in the UML package (block 67). If the answer to this enquiry is yes, then a return is make back to block 62 (FIG. 5B) as denoted by a connector E. If on the other hand, the answer to the above inquiry is no then a further inquiry is made to determine whether or not there are any more UML packages (block 68). If the answer to this inquiry is yes, then a return is made back to the block 61 (FIG. 5B) as denoted by a connector D. If on the other hand, the answer to the above inquiry is no, then a step of saving the tempfile as tempfile.txt is executed (block 69). After this, the process ends (bubble 70).") and (see Column 8, Lines 52-55, "when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the invention.").

As Per Claim 20, Stefaniak disclosed: A method for outputting data from a legacy computer system from a DOM instance as Extensible Markup Language, the method comprising:

-modifying an application of the legacy computer system to output data having a schema element; (see Fig. 6).

-generating data from the modified application; (see Column 2, Lines 33-36, "FIGS. 5A through 5C is a combined flow chart of the method for generating an UML/XML representation of the file reference model created by the method described with reference to FIG. 4 above.") and (see Column 6, Lines 56-57, "the answer to the above inquiry is no, then a step of saving the tempfile as tempfile.txt is executed (block 69).").

- aligning the schema element and the current context; (see Column 6, Lines 25-33, "Referring now to FIG. 5B at the connector A, an inquiry is made as to whether or not there are any more sub-projects in the file reference model of the project (diamond 58). If the answer to this inquiry is yes, then a return is made back to the block 52 (FIG. 5A) as denoted by a connector B. On the other hand, if the answer to this inquiry is no, then a command to create a "tempfile" is executed (block 59). Next the project name of the file reference model is tagged to this tempfile as a UML model name (block 60).").

-writing the output data schema element to a current one of plural contexts of an Extensible Markup Language schema; (see Column 6, Lines 59-67 to Column 7, Lines 1-4, "Referring now to FIG. 6 where a flowchart depicting the process of generating an XML file representation of the UML model created by the process described in FIG. 5A through 5C. The process begins with a start bubble 72, followed by a process of parsing the tempfile (block 73) for the UML model created in FIGS. 5A, 5B and 5C. Next, a call 74 is made to the various UML methods to build a model in memory. This is followed by a step 75 of reading the UML model from memory, created in the previous step, and building an XML file representation of the

model. Finally, the XML file representation created in the previous step is saved as <projectname.xml> (block 76) and the process ends (bubble 77).”). and

Stefaniak didn't explicitly disclose using a Document Object Model. However, van Elkeren taught populating a Document Object Model with the data to output an Extensible Markup Language instance. (see Column 12, Lines 5-10, “Preferably, a standard API, the XML DOM (Document Object Model) is used to access and manipulate XML data. The DOM provides a standard set of properties, methods, and events for program developers to use. This set of standards, the object model, provides a simple means of reading and writing data to and from an XML tree structure.”). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use DOM, as suggested by van Elkeren, to apply to output XML data. The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide a simple means of reading and writing data to and from an XML tree structure.

As Per Claim 21, the rejection of claims 20 is incorporated and further Stefaniak disclose aligning the schema element further comprises:

-determining that the schema element is a descendant of the current context; (see Column 6, Lines 25-33, “Referring now to FIG. 5B at the connector A, an inquiry is made as to whether or not there are any more sub-projects in the file reference model of the project (diamond 58). If the answer to this inquiry is yes, then a return is made back to the block 52 (FIG. 5A) as denoted by a connector B. On the other hand, if the answer to this inquiry is no, then a command to create

a "tempfile" is executed (block 59). Next the project name of the file reference model is tagged to this tempfile as a UML model name (block 60).”). and

-creating the Extensible Markup Language tags down through the schema element. (see Column 6, Lines 34-40, “This step is followed by the step of tagging the file reference sub-model name as a UML package name to the tempfile (block 61). Next, each screen specification in the UML package created in the previous step is parsed (block 62). Subsequently for each screen name, a UML class tag is assigned (block 63). The process illustration continues as denoted by a connector C.”)

5. Claims 10-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lection et al. (US Patent No. 6,418,446) hereafter Lection in view of Stefaniak (US Patent No. 6,550,054).

As Per Claim 10, the rejection of claims 7 is incorporated and further Lection disclosed:

-generating output data with an application; (see Column 3, Lines 48-52, “The selected record may be formatted as a first Document Object Model (DOM) tree, the gather verb specification may be formatted as a second DOM tree, and the output data destination may be formatted as a third DOM tree.”).

-outputting data from a Document Object Model instance from the writer engine according to the Extensible Markup Language schema. (see Column 3, Lines 8-12, “It is another object of the present invention to provide this technique using a DOM tree created from an XML

syntax representation of the source data, and creating an output DOM tree in which the destination data gathered from the source is reformatted and stored.").

Lection didn't explicitly disclose using a writer engine. However, Stefaniak taught calling a writer engine with the application; providing the generated output data to the writer engine. (see Column 5, Lines 43-57, "The terminal screens are discovered using the transform navigator 19, which produces application and screen specifications 32. The application and screen specifications 32 are then applied to the file warehouse 21, which produces the project file reference model 27. The model 27 is applied to the terminal-to-XML 20, which produces a UML model 34 in a MOF compliant repository. The UML model is then applied to the UML model to XMI/UML DTD generator 22, which produces XMI/UML DTD streams 35. The streams 35 may be used for several purposes, including transmitting legacy screen based application specifications over a network to modeling tools. From the modeling tools the application specifications could be viewed in an object oriented way compliant with the UML standard."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate XMI/UML DTD generator, as suggested by Stefaniak into the system of Lection, to produces XMI/UML DTD streams. The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide a simple means to generate XML codes.

As Per Claim 11, the rejection of claims 10 is incorporated and further Lection didn't explicitly disclose a legacy computer system application. However, Stefaniak teaches a legacy computer system application (see Column 1, Lines 58-67, "These and other objects, which will

become apparent as the invention is described in detail below, are provided in a computer-implemented method that automatically converts text-based screen applications of a legacy computer system into a graphical-based representation thereof. The method includes the steps of transforming a terminal-based screen application into an application specification; converting the application specification into a modeling language-based representation; and, displaying the modeling language-based representation with a graphical user interface."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use legacy computer system, as suggested by Stefaniak , to build DOM instance of Lection. The modification would have been obvious because one of ordinary skill in the art would have been motivated to make application run on more computer systems.

As Per Claim 12, the rejection of claims 11 is incorporated and further Lection didn't explicitly disclose the writer engine comprises an application run in the computer language of the legacy computer system application. However, Stefaniak teaches the writer engine comprises an application run in the computer language of the legacy computer system application (see Column 6, Lines 41-58, "Referring now to FIG. 5C, at the connector C, a UML attribute and a UML datatype tag are assigned for each field name associated with the screen (block 65). Next, all the operations of the screen are initialized and transmitted as the operations of the UML class created for that screen (block 66). This is followed by making an inquiry whether or not there are any more screens in the UML package (block 67). If the answer to this enquiry is yes, then a return is make back to block 62 (FIG. 5B) as denoted by a connector E. If on the other hand, the answer to the above inquiry is no then a further inquiry is made to determine whether or not there

are any more UML packages (block 68). If the answer to this inquiry is yes, then a return is made back to the block 61 (FIG. 5B) as denoted by a connector D. If on the other hand, the answer to the above inquiry is no, then a step of saving the tempfile as tempfile.txt is executed (block 69). After this, the process ends (bubble 70).") and (see Column 8, Lines 52-55, "when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the invention."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use application in computer language, as suggested by Stefaniak, to run on legacy computer system. The modification would have been obvious because one of ordinary skill in the art would have been motivated to make application more flexible to run in different computer systems.

As Per Claim 13, Lection disclosed: A system for outputting data from a Document Object Model as Extensible Markup Language, the system comprising:

- a computer system having an application that outputs data; (see Column 3, Lines 48-52, "The selected record may be formatted as a first Document Object Model (DOM) tree, the gather verb specification may be formatted as a second DOM tree, and the output data destination may be formatted as a third DOM tree.").

Lection didn't explicitly disclose using a writer engine. However, Stefaniak taught a engine operable to write the output data in plural active contexts; wherein the application calls the writer engine when the application outputs data, the writer engine operable to build a Document Object Model instance for output of the data in accordance with the Extensible

Markup Language schema. (see Column 5, Lines 43-57, "The terminal screens are discovered using the transform navigator 19, which produces application and screen specifications 32. The application and screen specifications 32 are then applied to the file warehouse 21, which produces the project file reference model 27. The model 27 is applied to the terminal-to-XML 20, which produces a UML model 34 in a MOF compliant repository. The UML model is then applied to the UML model to XMI/UML DTD generator 22, which produces XMI/UML DTD streams 35. The streams 35 may be used for several purposes, including transmitting legacy screen based application specifications over a network to modeling tools. From the modeling tools the application specifications could be viewed in an object oriented way compliant with the UML standard."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate XMI/UML DTD generator, as suggested by Stefaniak into the system of Lection, to produces XMI/UML DTD streams. The modification would have been obvious because one of ordinary skill in the art would have been motivated to provide a simple means to generate XML codes.

As Per Claim 14, the rejection of claims 13 is incorporated and further Lection didn't explicitly disclose aligned element. However, Stefaniak taught the writer engine populates a Document Object Model as a schema element aligned with the current one of the contexts by creating Extensible Markup Language tagged nodes down through the schema element of the output data if the schema element of the output data is a descendant of the current context. (see Column 6, Lines 25-33, "Referring now to FIG. 5B at the connector A, an inquiry is made as to whether or not there are any more sub-projects in the file reference model of the project

(diamond 58). If the answer to this inquiry is yes, then a return is made back to the block 52 (FIG. 5A) as denoted by a connector B. On the other hand, if the answer to this inquiry is no, then a command to create a "tempfile" is executed (block 59). Next the project name of the file reference model is tagged to this tempfile as a UML model name (block 60).") and (see Column 6, Lines 41-58, "Referring now to FIG. 5C, at the connector C, a UML attribute and a UML datatype tag are assigned for each field name associated with the screen (block 65). Next, all the operations of the screen are initialized and transmitted as the operations of the UML class created for that screen (block 66). This is followed by making an inquiry whether or not there are any more screens in the UML package (block 67). If the answer to this enquiry is yes, then a return is make back to block 62 (FIG. 5B) as denoted by a connector E. If on the other hand, the answer to the above inquiry is no then a further inquiry is made to determine whether or not there are any more UML packages (block 68). If the answer to this inquiry is yes, then a return is made back to the block 61 (FIG. 5B) as denoted by a connector D. If on the other hand, the answer to the above inquiry is no, then a step of saving the tempfile as tempfile.txt is executed (block 69). After this, the process ends (bubble 70).."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Stefaniak into the system of Lection, to align the elementas. The modification would have been obvious because one of ordinary skill in the art would have been motivated to easily process the preformatted documents.

6. Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Stefaniak (US Patent No. 6,550,054) in view of van Elkeren et al. (US Patent No. 6,618,852) hereafter van Elkeren further in view of Sandhu et al. (US Patent No. 6,347,307) hereafter Sandhu.

As Per Claim 3, the rejection of claim 1 is incorporated and further Stefaniak and van Elkeren taught written using a Document Object Model (see van Elkeren , Column 12, Lines 5-10, "Preferably, a standard API, the XML DOM (Document Object Model) is used to access and manipulate XML data. The DOM provides a standard set of properties, methods, and events for program developers to use. This set of standards, the object model, provides a simple means of reading and writing data to and from an XML tree structure."). Stefaniak and van Elkeren didn't explicitly disclose applying XSLT stylesheets. However, Sandhu taught applying one or more XSLT stylesheets to restructure the Document Object Model instance for outputting data in a predetermined format. (see Column 50, Lines 24-30, "Next, the Connect Processor invokes a XSLT processor 1450--an off-the-shelf component (e.g., International Business Machines Corp.'s "Alphaworks")--to apply the rules of the XSL stylesheet 1440 to DOM tree 1430 (step 1520). This process results in the generation of a FinXML document 1460 (step 1530) that can be used by the CFOWeb System."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use XSLT stylesheets, as suggested by Sandhu, to apply to DOM. The modification would have been obvious because one of ordinary skill in the art would have been motivated to make the document usable by other computer system.

7. Claims 15-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lection et al. (US Patent No. 6,418,446) hereafter Lection in view of Stefaniak (US Patent No. 6,550,054) further in view of Shanmugasundaram et al. "Relational Databases for Querying XML Documents: Limitations and Opportunities". Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, hereafter Shanmugasundaram.

As Per Claim 15, the rejection of claims 14 is incorporated and further Lection and Stefaniak didn't explicitly disclose traverse the Extensible Markup Language tagged nodes for the current context up to the minimal mutual ancestor. However, Shanmugasundaram taught the writer engine is further operable to determine a minimal mutual ancestor of the schema element and the current context and to traverse the Extensible Markup Language tagged nodes for the current context up to the minimal mutual ancestor and to create Extensible Markup Language tags for the schema element down from the mutual ancestor. (see page 5, left hand column last 2 lines to right hand column, lines 1-14, "Do a depth first traversal of the DTD graph, starting at the element node for which we are constructing relations. Each node is marked as "visited" the first time it is reached and is unmarked it once all its children have been traversed. If an unmarked node in the DTD graph is reached during depth first traversal, a new node bearing the same name is created in the element graph. In addition, a regular edge is created from the most recently created node in the element graph with the same name as the DFS parent of the current DTD node to the newly created node. If an attempt is made to traverse an already marked DTD node, then a backpointer edge is added from the most recently created node in the element graph to the most recently created node in the element graph with the same name as the marked DTD

node."), and (see page 6, left hand column last 2 lines to right hand column first three lines, "Finally, of the mutually recursive elements all having indegree one (such as monograph and editor in Figure 8), one of them is made a separate relation. We can find such mutually recursive elements by looking for strongly connected components in the DTD graph."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use traversed technique, as suggested by Sandhu, to construct DTD node. The modification would have been obvious because one of ordinary skill in the art would have been motivated to lookup node in DTD graph.

As Per Claim 16, the rejection of claims 13 is incorporated and further Lection and Stefaniak disclose the computer system comprises a legacy computer system. (see Stefaniak , Column 1, Lines 58-67, "These and other objects, which will become apparent as the invention is described in detail below, are provided in a computer-implemented method that automatically converts text-based screen applications of a legacy computer system into a graphical-based representation thereof. The method includes the steps of transforming a terminal-based screen application into an application specification; converting the application specification into a modeling language-based representation; and, displaying the modeling language-based representation with a graphical user interface.").

As Per Claim 17, the rejection of claims 16 is incorporated and further Lection and Stefaniak disclose the application comprises a legacy computer system application modified to

output an Extensible Markup Language schema element with output data. (see Stefaniak, Column 5, Lines 43-57, "The terminal screens are discovered using the transform navigator 19, which produces application and screen specifications 32. The application and screen specifications 32 are then applied to the file warehouse 21, which produces the project file reference model 27. The model 27 is applied to the terminal-to-XML 20, which produces a UML model 34 in a MOF compliant repository. The UML model is then applied to the UML model to XMI/UML DTD generator 22, which produces XMI/UML DTD streams 35. The streams 35 may be used for several purposes, including transmitting legacy screen based application specifications over a network to modeling tools. From the modeling tools the application specifications could be viewed in an object oriented way compliant with the UML standard.").

As Per Claim 18, the rejection of claims 16 is incorporated and further Lection and Stefaniak disclose the writer engine is written in the code of the legacy computer system. (see Stefaniak , Column 6, Lines 41-58, "Referring now to FIG. 5C, at the connector C, a UML attribute and a UML datatype tag are assigned for each field name associated with the screen (block 65). Next, all the operations of the screen are initialized and transmitted as the operations of the UML class created for that screen (block 66). This is followed by making an inquiry whether or not there are any more screens in the UML package (block 67). If the answer to this enquiry is yes, then a return is make back to block 62 (FIG. 5B) as denoted by a connector E. If on the other hand, the answer to the above inquiry is no then a further inquiry is made to determine whether or not there are any more UML packages (block 68). If the answer to this inquiry is yes, then a return is made back to the block 61 (FIG. 5B) as denoted by a connector D.

If on the other hand, the answer to the above inquiry is no, then a step of saving the tempfile as tempfile.txt is executed (block 69). After this, the process ends (bubble 70).") and (see Column 8, Lines 52-55, "when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the invention.").

8. Claim 19 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lection et al. (US Patent No. 6,418,446) hereafter Lection in view of Stefaniak (US Patent No. 6,550,054) further in view of Shanmugasundaram et al. "Relational Databases for Querying XML Documents: Limitations and Opportunities". Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, hereafter Shanmugasundaram, further in view of Vermeire el al. (US Patent No. 6,209,124) hereafter Vermeire.

As Per Claim 19, the rejection of claims 18 is incorporated and further Lection, Stefaniak and Shanmugasundaram didn't explicitly disclose cobol. However, Vermeire taught *the code comprises COBOL.* (see Column 18, Lines 18-21, "Alternatively, the XML document of Table 20 could have been generated from COBOL source code which would have appeared as in Table 22."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use COBOL, as suggested by Vermeire, as the application language. The modification would have been obvious because one of ordinary skill in the art would have been motivated to utilizes a constructed intermediary acting with the host machine and its program applications. (see Column 5, Lines 7-17, "The invention utilizes a constructed intermediary which is user defined based upon the application language utilized by the host

computer. The intermediary is further constructed to encompass the machine architecture and data structures involved in the host machine and application programs. This then allows the intermediary to function to restructure in-memory binary data streams received from the host into XML documents and to restructure XML documents into binary data streams capable of acting with the host machine and its program applications.").

9. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Stefaniak (US Patent No. 6,550,054) in view of van Elkeren et al. (US Patent No. 6,618,852) hereafter van Elkeren, further in view of Shanmugasundaram et al. "Relational Databases for Querying XML Documents: Limitations and Opportunities". Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999, hereafter Shanmugasundaram.

As Per Claim 22, the rejection of claims 21 is incorporated and further Stefaniak and van Elkeren disclosed

- determining a minimal mutual ancestor of the schema element and the current context; (see Stefaniak , Column 6, Lines 25-33, "Referring now to FIG. 5B at the connector A, an inquiry is made as to whether or not there are any more sub-projects in the file reference model of the project (diamond 58). If the answer to this inquiry is yes, then a return is made back to the block 52 (FIG. 5A) as denoted by a connector B. On the other hand, if the answer to this inquiry is no, then a command to create a "tempfile" is executed (block 59). Next the project name of the file reference model is tagged to this tempfile as a UML model name (block 60)."). and (see Column 6, Lines 25-33, "Next, an inquiry (block 55) is made to determine whether or not there

are any more screen specifications in the file reference model of the sub-project. If the answer to this inquiry is yes, then a return is made back to the block 53. On the other hand, if the answer to this inquiry is no, then the process illustration continues in FIG. 5B as denoted by a connector A."). "diamond 58" and "block 55" determine the minimal mutual ancestor. and

- creating the Extensible Markup Language tags for the schema element down from the mutual ancestor. (see Stefaniak , Column 6, Lines 34-40, "This step is followed by the step of tagging the file reference sub-model name as a UML package name to the tempfile (block 61). Next, each screen specification in the UML package created in the previous step is parsed (block 62). Subsequently for each screen name, a UML class tag is assigned (block 63). The process illustration continues as denoted by a connector C.")

Stefaniak and van Elkeren didn't explicitly disclose traverse the Extensible Markup Language tagged nodes for the current context up to the minimal mutual ancestor. However, Shanmugasundaram taught traversing the Extensible Markup Language tags for the current context up to the mutual ancestor (see page 5, left hand column last 2 lines to right hand column, lines 1-14, "Do a depth first traversal of the DTD graph, starting at the element node for which we are constructing relations. Each node is marked as "visited" the first time it is reached and is unmarked it once all its children have been traversed. If an unmarked node in the DTD graph is reached during depth first traversal, a new node bearing the same name is created in the element graph. In addition, a regular edge is created from the most recently created node in the element graph with the same name as the DFS parent of the current DTD node to the newly created node. If an attempt is made to traverse an already marked DTD node, then a backpointer edge is added

from the most recently created node in the element graph to the most recently created node in the element graph with the same name as the marked DTD node."), and (see page 6, left hand column last 2 lines to right hand column first three lines, "Finally, of the mutually recursive elements all having indegree one (such as monograph and editor in Figure 8), one of them is made a separate relation. We can find such mutually recursive elements by looking for strongly connected components in the DTD graph."). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use traversed technique, as suggested by Sandhu, to construct DTD node. The modification would have been obvious because one of ordinary skill in the art would have been motivated to lookup node in DTD graph.

Response to Arguments

10. Applicant's arguments with respect to claims 1-23 and 25 have been considered but they are not persuasive.

In the remarks, the applicant primarily argues that:

i) As per claim 23, Applicant acknowledge that Stefaniak teaches the output of the system is a representation or model of the terminal-based application. Applicant argues "there is no modification of the terminal-based application in Stefaniak" (Remark page 9, lines 11-13).

ii) As per Claim 7, Applicant's characterizing of Lectio's teaching that fails to address

a) establishing a relationship of one ... Extensible Markup Language Document Object Model context. (Remark page 11, lines 4-5)

b) building a Document Object Model instance with the one... contexts. (Remark page 11, lines 6 & 18-19)

Examiner's response:

i) Examiner disagrees with applicant's assertion that "there is no modification of the terminal-based application in Stefaniak". Stefaniak discloses a system that describes legacy application screens in term of a terminal application specification and converts the specification into a UML model (E.g. see Abstract and col. 1:57-67). Stefaniak also discloses using the transform navigator 19, which produces application and screen specification. (E.g. see col. 5:41-45). As showing in FIG. 4, one of the transform processes is the project 46, which is "create/edit file reference model for project" that produce the final result display 48 for an end user 49 to view. (Note that edit is known as "to alter, adapt"; modify is known as "to make change"; convert is known as "to change from from one form or function to another"; transform is known as "to change in composition or structure", see "Webster's II New Riverside University Dictionary", The Riverside Publishing Company, pages 418,762,308,1226 respectively).

ii) Examiner disagree with how the Applicant's characterizing of Lectio's teaching.

a) As pointed out in previous Office Action (pages 3-4) and as noted above in claim 7, DOM tree is the relationship of the output data. The relationship must have been established in the DOM tree, otherwise the DOM tree will be inoperable (E.g. see FIG. 3D and associated text.).

b) Examiner interprets the representation of the selected record and/or representation of the gather verb specification is context, as set forth above in Claim 7. Thus, arguments are moot and not persuasive, and the claims are stayed finally rejected as set forth above.

Conclusion

11. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Correspondence Information

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Kuo-Liang J Tang whose telephone number is 703-305-4866. The examiner can normally be reached on M-F 8:30 to 5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q Dam can be reached on 703-305-4552.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306.

Kuo-Liang J. Tang

Software Engineer Patent Examiner

WEI Y. ZHEN
PRIMARY PATENT EXAMINER